МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Кемеровский государственный университет» (КемГУ)

**Институт Цифры**

**Кафедра цифровых технологий**

**ОТЧЕТ**

**О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ №4(4)**

**ПО ПРОГРАММНОЙ ИНЖЕНЕРИИ**

Студента 3 курса, группы ПИ-202

Балахонцева Владимира Игоревича

Направление подготовки

*09.03.03 Прикладная информатика*

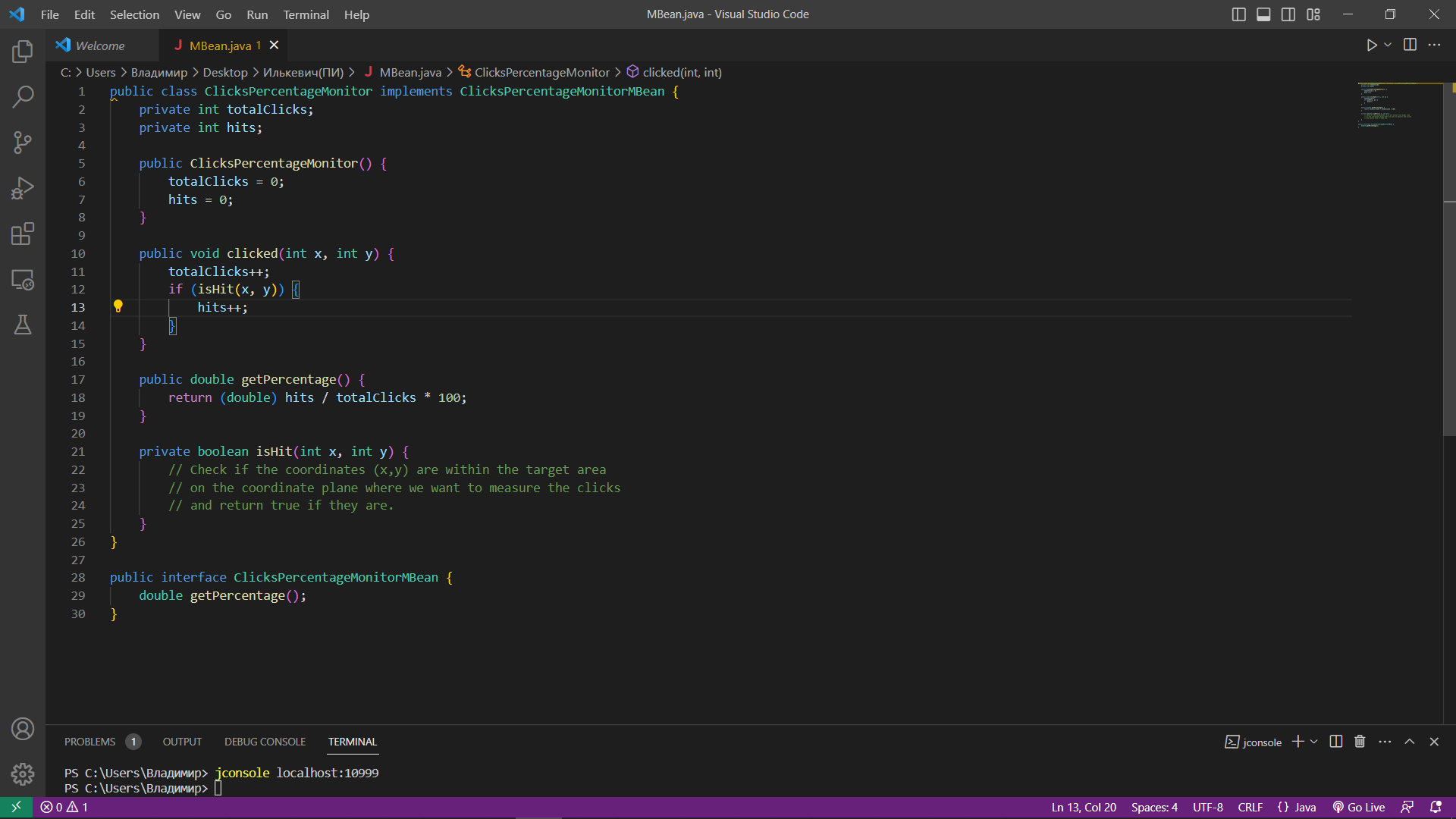
Руководитель:

ассистент

В.В. Илькевич

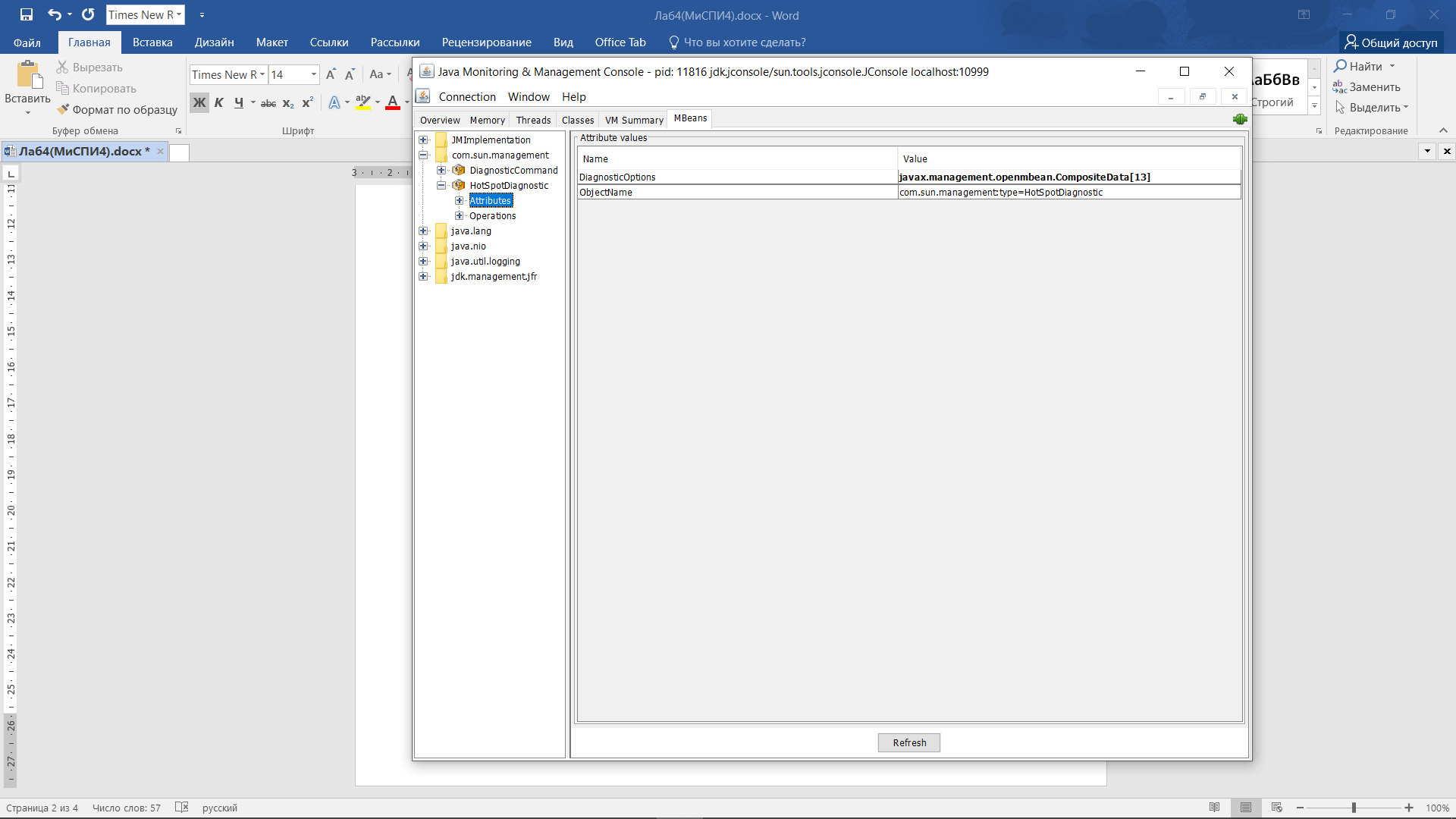
Кемерово 2023

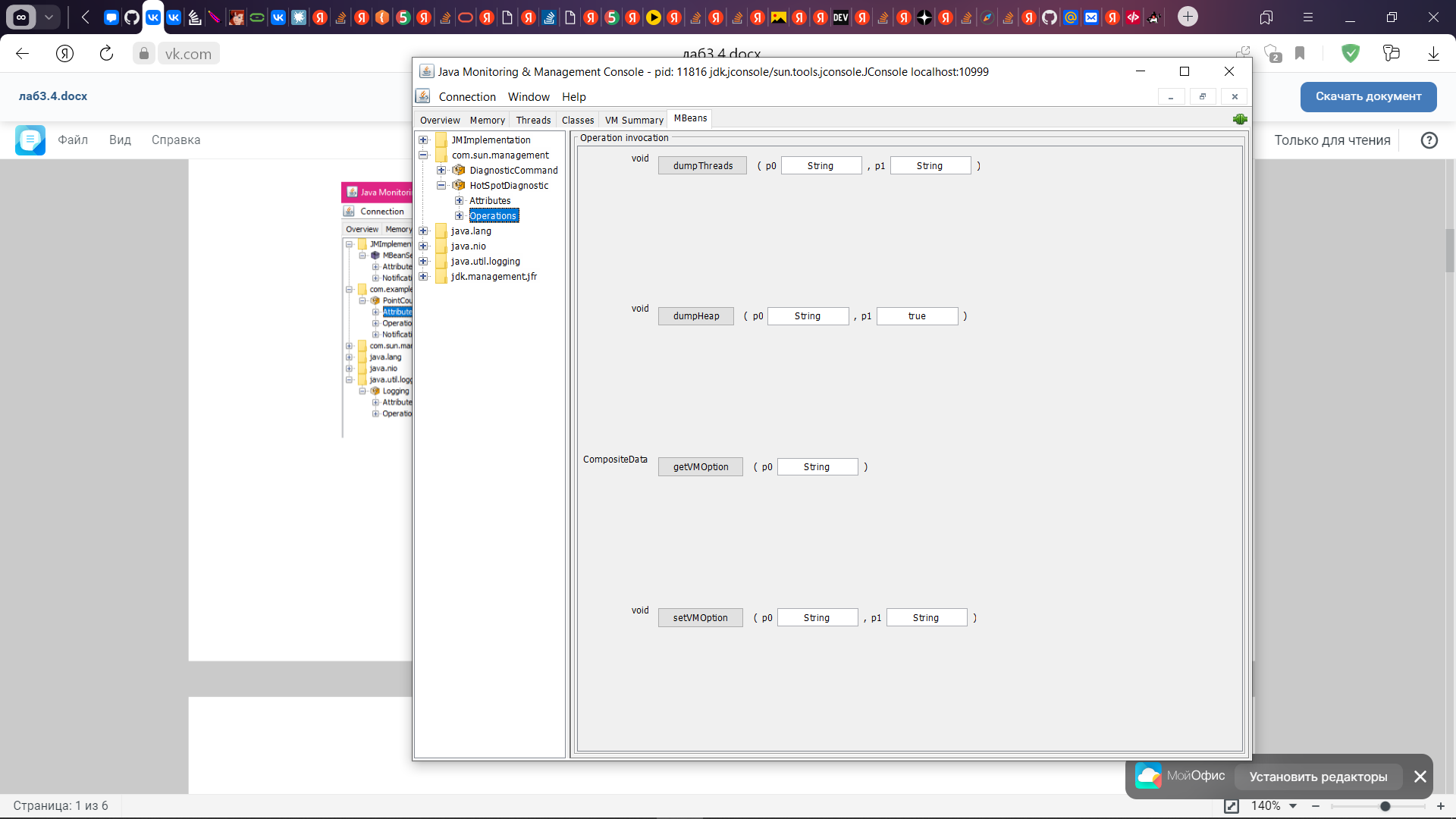
MBean пример кода:

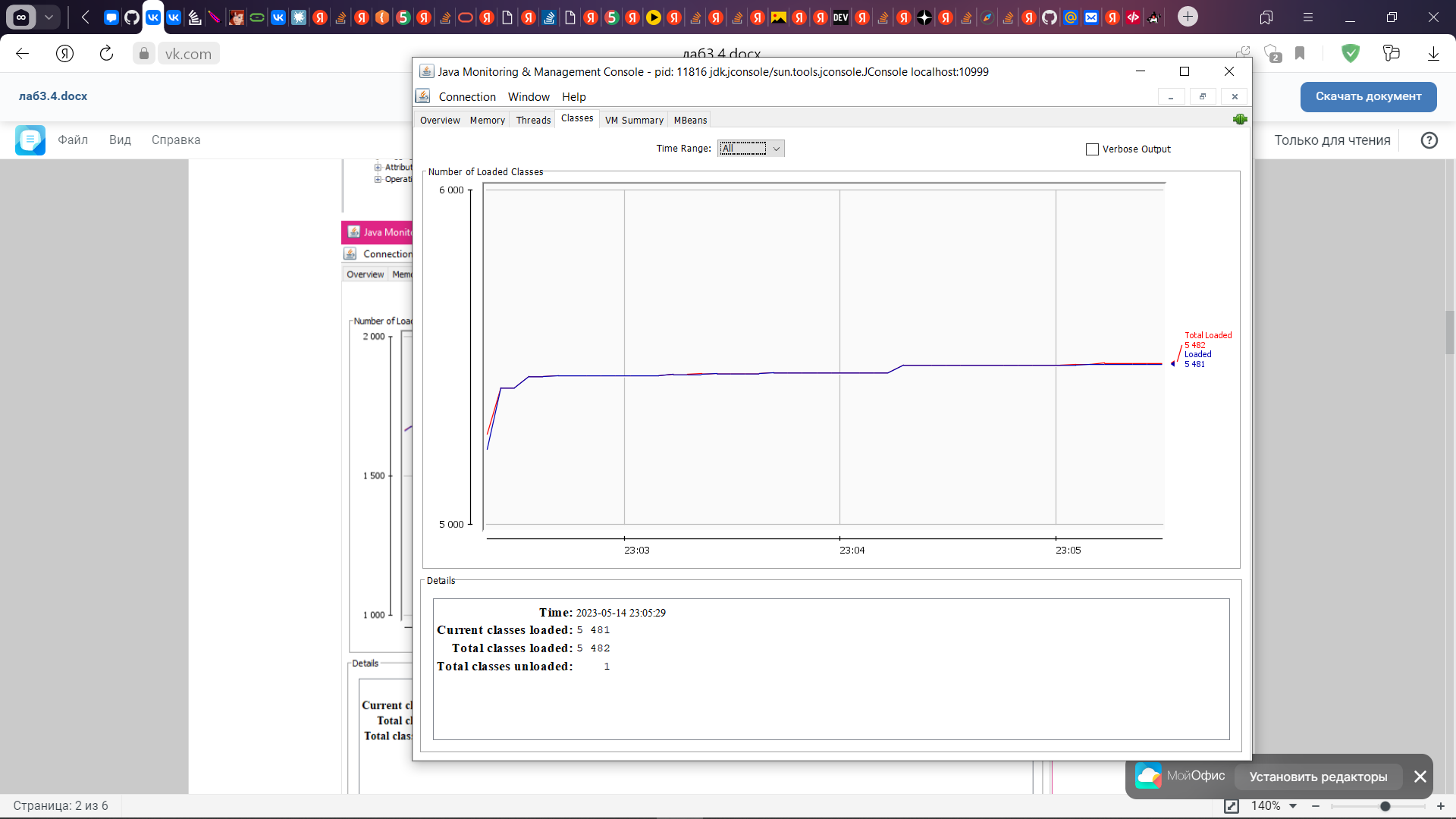


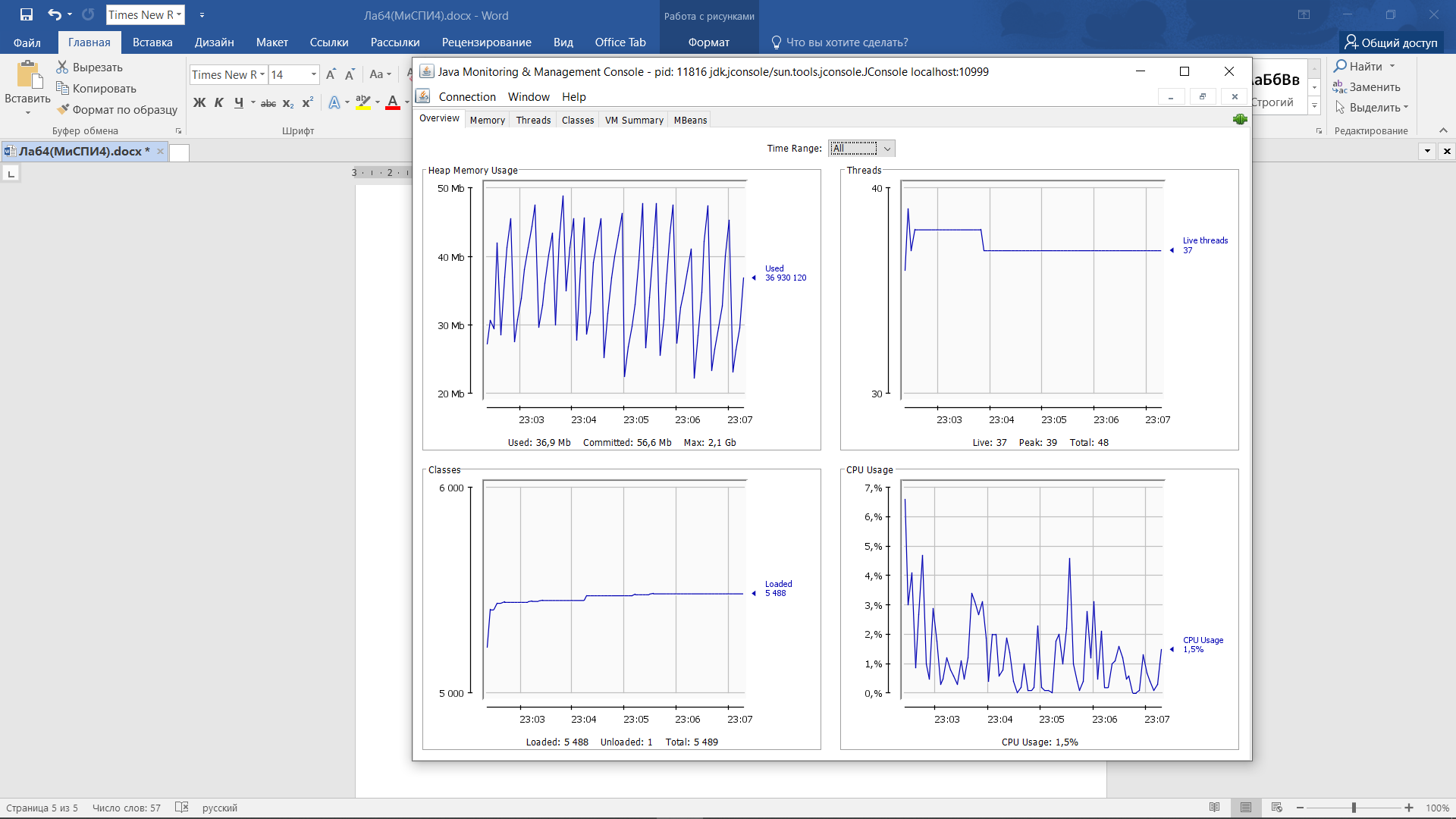
**Показания MBean-классов:**

**2. JConsole**

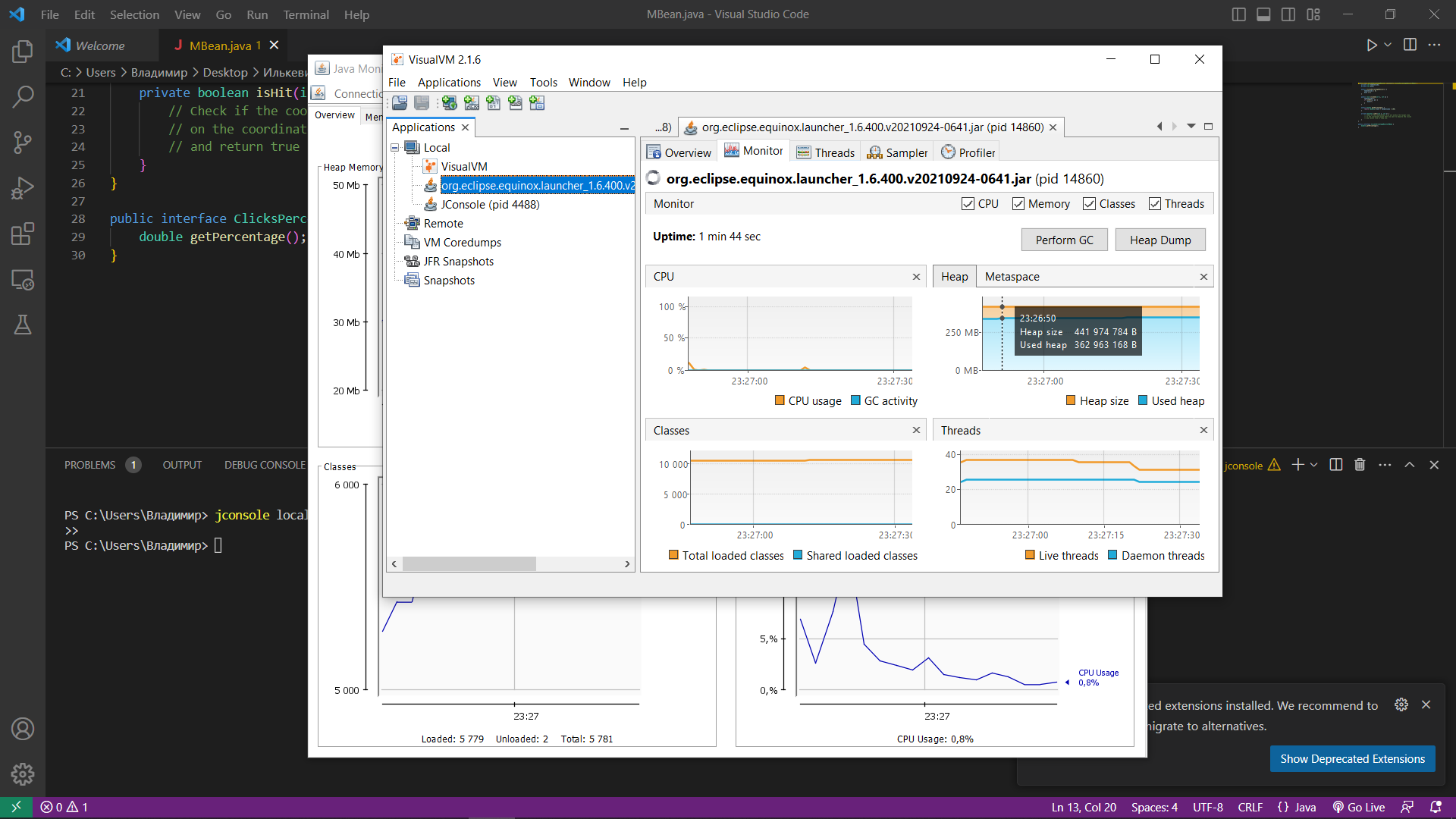


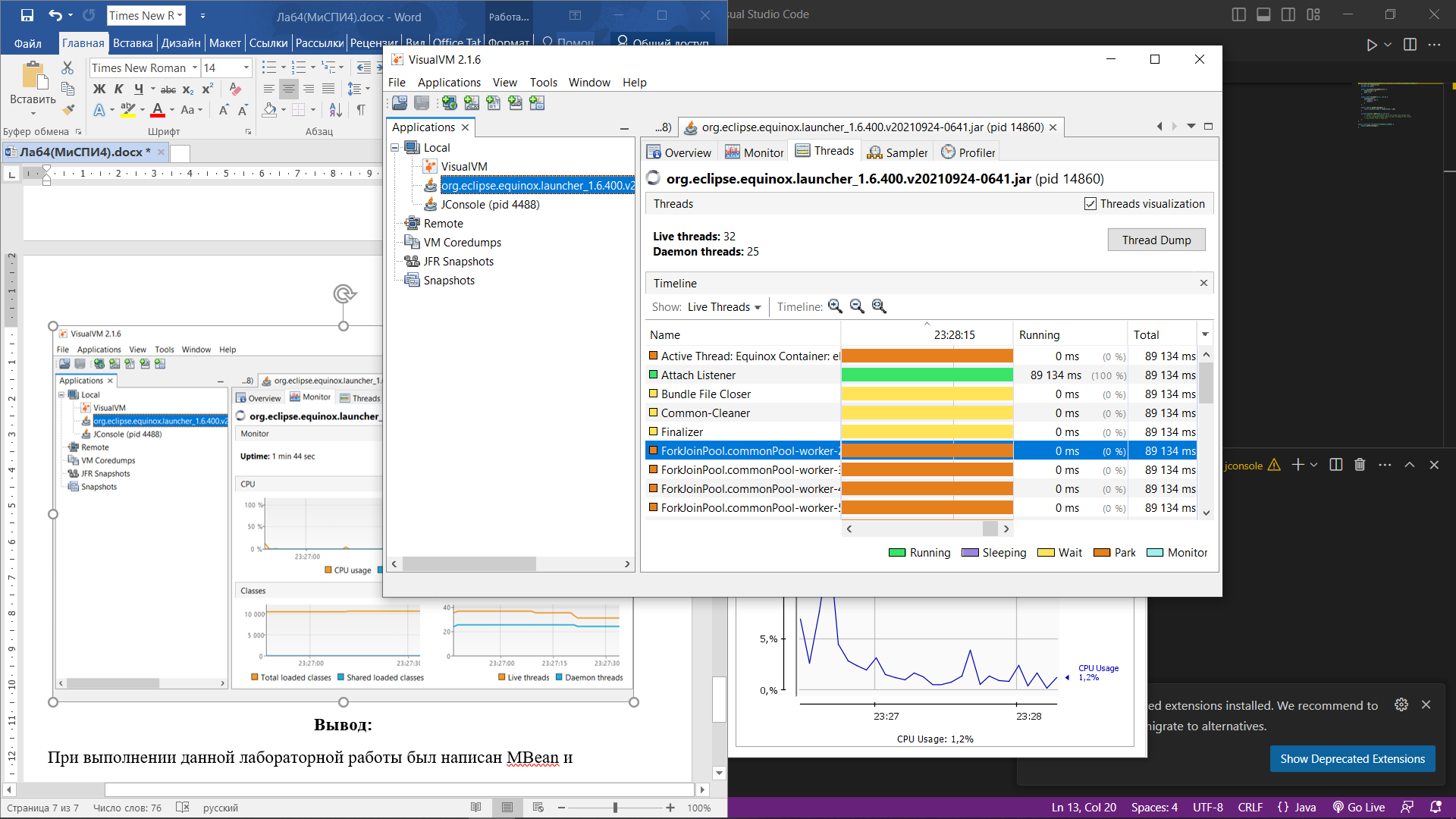


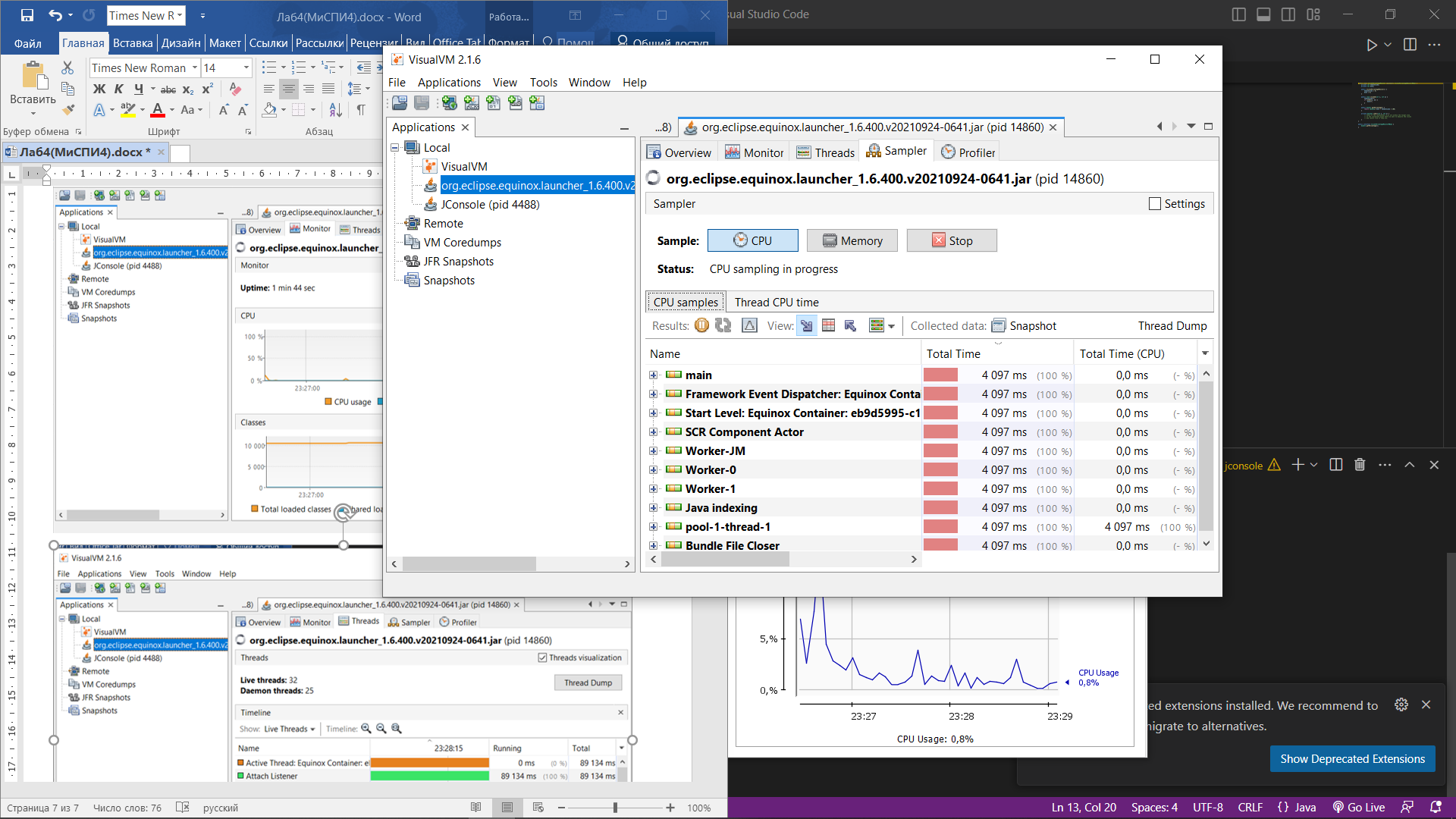


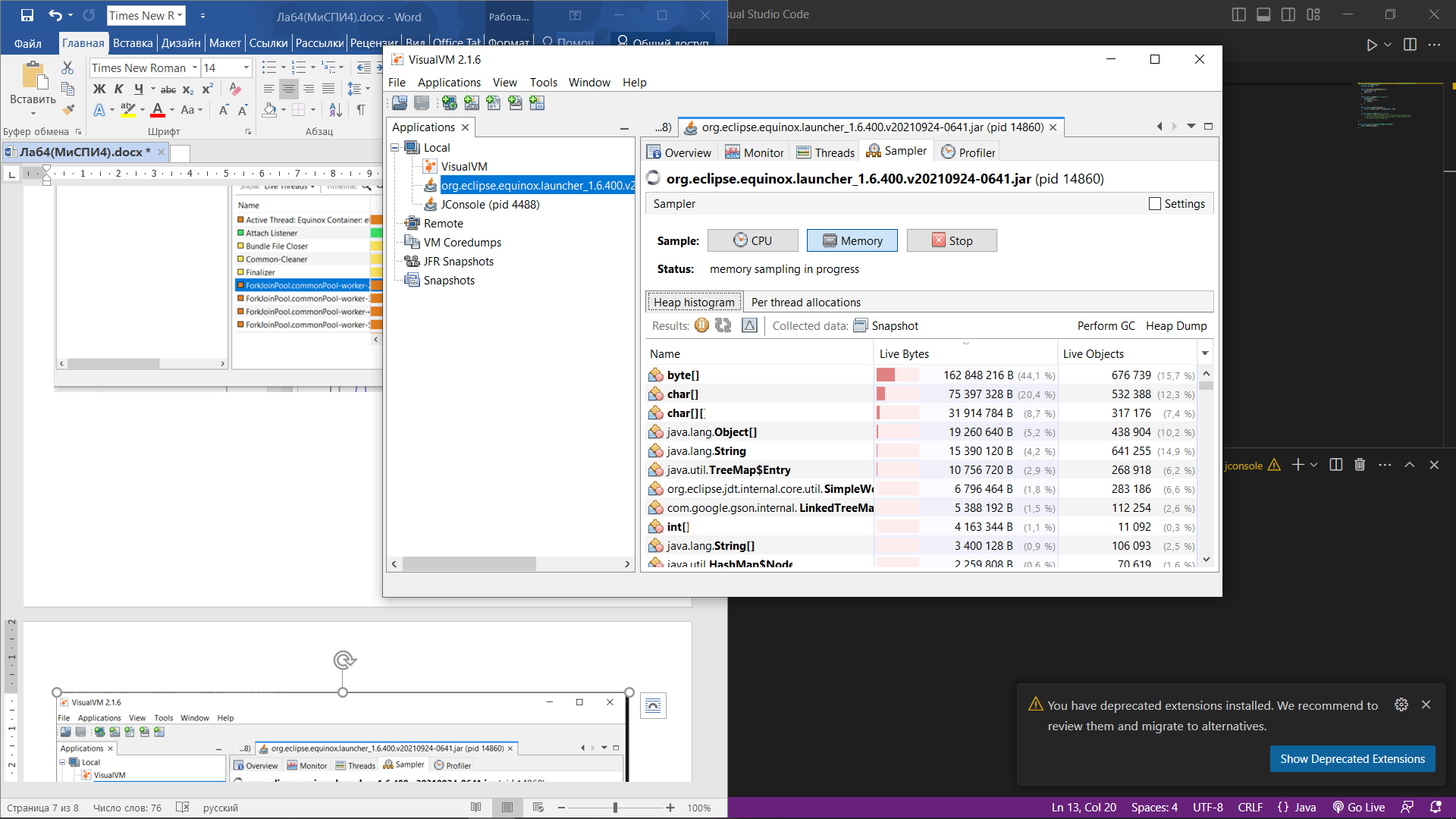


**3. VisualVM**









**Вывод:**

При выполнении данной лабораторной работы был написан MBean и

проведено ознакомление с JConsole и VisualVM

**Вопросы:**

1. Мониторинг и профилирование. Основные понятия. Отличия мониторинга от профилирования.

Мониторинг — система постоянного наблюдения за явлениями и процессами, проходящими в окружающей среде и обществе, результаты которого служат для обоснования управленческих решений по обеспечению безопасности людей и объектов экономики.

Профилирование — это сбор характеристик программы во время ее выполнения. При профилировании замеряется время выполнения и количество вызовов отдельных функций и строк в коде программы. При помощи этого инструмента программист может найти наиболее медленные участки кода и провести их оптимизацию.

Мониторинг = отслеживание того, как запущено приложение (использование памяти, процессора, диска и т.д.). Профилирование = измерение производительности (скорости) путем запуска кода и определения времени, которое занимает каждый шаг.

2. Инфраструктура для организации мониторинга и профилирования в составе JDK. JMX.

Начиная с 7u40 в составе Oracle JDK появился GUI инструмент для реалтайм мониторинга и профилирования JVM. Он позволяет собрать очень много информации о JVM процессе, найти узкие места и проблемы, которые влияют на производительность.

Mission Control включает в себя JMX консоль и Java Flight Recorder (JFR). С помощью JMX осуществяется взаимодействие с удаленным Java процессом, а JFR собирает данные о событиях.

3. MBeans. Основные понятия. Архитектура фреймворка.

MBEANS технологии JMX - это managed beans, а именно объекты Java, которые представляют ресурсы, подлежащие управлению. MBean имеет management interface, состоящий из следующего:

• Именованные и типизированные атрибуты, которые могут быть прочитаны и записаны.

• Именованные и типизированные операции, которые могут быть вызваны.

• Типизированные уведомления, которые могут быть отправлены MBean.

Платформа MXBean - это MBean для мониторинга виртуальной машины Java и других компонентов среды выполнения Java и управления ими. Каждый MXBean инкапсулирует часть функциональных возможностей виртуальной машины, таких как система загрузки классов, система компиляции точно в срок (JIT), сборщик мусора.

Каждый экземпляр WebLogic владеет сервером MBean, на котором размещено несколько MBeans. Сервер MBean действует как реестр для MBeans и предоставляет службы для доступа к MBeans, запущенным на сервере, и управления ими. Поскольку домен WebLogic может быть распределен по нескольким компьютерам с различными развертываниями и ресурсами домена, предназначенными для разных серверов, сервер MBean для каждого экземпляра WebLogic будет содержать разные типы MBeans. Например, статистику времени выполнения для пула подключений JDBC на сервере можно получить только из MBean, запущенного на этом сервере.

4. Утилита JConsole. Возможности, область применения.

Графический пользовательский интерфейс JConsole - это инструмент мониторинга, который соответствует спецификации Java Management Extensions (JMX). JConsole использует обширный инструментарий виртуальной машины Java (Java VM) для предоставления информации о производительности и потреблении ресурсов приложениями, работающими на платформе Java. Можно легко использовать JConsole (или его ближайший родственник VisualVM) для мониторинга производительности приложений Java и отслеживания кода в Java.

5. Утилита Visual VM. Возможности, область применения.

Java VisualVM - это инструмент, который предоставляет визуальный интерфейс для просмотра подробной информации о приложениях, основанных на технологии Java (Java applications), во время их запуска на виртуальной машине Java (JVM). Java VisualVM организует данные JVM, которые извлекаются средствами Java Development Kit (JDK), и представляет информацию таким образом, чтобы вы могли быстро просматривать данные в нескольких приложениях Java. Вы можете просматривать данные о локальных приложениях и приложениях, запущенных на удаленных хостах. Вы также можете собирать данные о программном обеспечении JVM и сохранять их в своей локальной системе, а затем просматривать данные позже или делиться ими с другими пользователями.

Если у вас уже есть приложение, которое надо оптимизировать — то как пользоваться этими инструментами более-менее понятно.

Если же ваша цель более академическая — то вам понадобится какая-то программа на Java. Начать исследовать можно с этой, сейчас она позволяет создавать в памяти куски памяти и «забывать» о них (создавая «мусор»).

Однако, чтобы хорошо поиграть с виртуальной машиной, надо побольше узнать об особенностях ее работы. Например, можно исследовать следующее:

• написать программу, нарушающую «гипотезу о потоковой локальности» (ведь JVM считает, что большинство объектов используются только одним потоком);

• работу с объектами разного размера — в JVM есть настройки, позволяющие большим объектам сразу попадать в Permanent generation;

• освобождение памяти из-под объектов с финализаторами. Не секрет, что JVM не спешит удалять их, но можно также поместить в финализаторы код, требующий длительных вычислений;

• использование гипотезы о том, что «объекты умирают молодыми».

• работу JIT-компилятора — постоянно перестраивать common path;li>

• написать очень большую программу (скорее всего сгенерировать), исследовать сколько памяти расходуется при оптимизации кода.

6. Удалённый мониторинг и профилирование приложений на платформе Java.

Профилировщик Java - это инструмент, который отслеживает конструкции и операции байт-кода Java на уровне JVM. Эти конструкции и операции кода включают создание объекта, итеративные выполнения (включая рекурсивные вызовы), выполнение методов, выполнение потоков и сборку мусора.

JProfiler - лучший выбор для многих разработчиков. Благодаря интуитивно понятному пользовательскому интерфейсу JProfiler предоставляет интерфейсы для просмотра производительности системы, использования памяти, потенциальных утечек памяти и профилирования потоков.

YourKit обладает основными функциями для визуализации потоков, сборки мусора, использования памяти и утечек памяти с поддержкой локального и удаленного профилирования через ssh-туннелирование.

Java VisualVM - это упрощенный, но надежный инструмент профилирования для приложений Java. Это бесплатный профилировщик с открытым исходным кодом. Этот инструмент входил в комплект Java Development Kit (JDK) вплоть до JDK 8, но был удален в JDK 9 и теперь распространяется как отдельный инструмент: загрузка VisualVM. Его работа зависит от других автономных инструментов, предоставляемых в JDK, таких как JConsole, jstat, jstack, jinfo и jmap.

Профилировщик NetBeans поставляется в комплекте с IDE Oracle NetBeans с открытым исходным кодом. Хотя этот профилировщик имеет много общего с Java VisualVM, это хороший выбор, когда мы хотим, чтобы все было объединено в одной программе (IDE + Profiler).

IntelliJ Profiler - это простой, но мощный инструмент для профилирования распределения ресурсов процессора и памяти. Он сочетает в себе возможности двух популярных профилировщиков Java: JFR и Async profiler.

Несмотря на наличие некоторых расширенных функций, основное внимание уделяется простоте использования. IntelliJ Profiler позволяет нам начать работу в несколько кликов без какой-либо настройки, предоставляя полезные функции, помогающие в нашей повседневной работе по разработке.